SIMULADOR PARA REDES DE PETRI HÍBRIDAS

J. M. Maestre
pepemaestre@cartuja.us.es

E.F. Camacho
eduardo@cartuja.us.es

Resumen

El siguiente artículo presenta un programa informático para la simulación de redes de Petri. A pesar de la gran cantidad de simuladores ya existentes, el aquí presentado posee una serie de características únicas que lo convierten en una herramienta de enorme utilidad para el ingeniero. El simulador está orientado a la simulación de sistemas híbridos de manera integrada: todos los componentes de las redes son híbridos, de manera que se rompe con el enfoque tradicional de integrar componentes de naturaleza discreta y continua para el modelado de fenómenos híbridos. Otra característica de interés es la posibilidad de emplear como coste asociado a las transiciones expresiones en función de magnitudes ya definidas en el sistema, del tiempo o incluso de variables aleatorias, lo que proporciona gran flexibilidad de modelado. Por último, el programa permite su integración de forma sencilla con MATLAB ©mediante DDE (Dynamic Data Exchange).

Palabras clave: Simulador Redes de Petri, Sistemas Híbridos, Sistemas Estocásticos, Matlab.

1 INTRODUCCIÓN

La problemática que envuelve el estudio de los sistemas híbridos precisa herramientas de modelado y simulación que simplifiquen las labores de diseño y testeo de sus sistemas de control. Por desgracia, la mayoría de las aplicaciones existentes, como el toolbox de redes de Petri de Matlab, están poco orientados a tareas de control propiamente dichas.

En los últimos años se han definido un gran número de formalismos para representar sistemas híbridos. Algunos de ellos han sido definidos ad hoc para recoger el fenómeno híbrido, mientras que otros han aprovechado la mayor penetración dentro de los círculos científicos de ciertos paradigmas de modelado a los que han añadido diversas extensiones con el mismo objetivo. El lenguaje de modelado que se emplea como punto de partida en esta tarea es el de las redes de Petri. En concreto, el programa que presentamos en este artículo sigue la variante de las redes de Petri presentada por R. Drath en [3], en la que se aplican algunos conceptos de orientación a objetos al paradigma híbrido de las redes de Petri, es decir, se asocia a cada elemento de la red una cierta estructura con campos de información que la caracterizan.

Se asume que el lector dispone de conocimientos básicos sobre redes de Petri. En caso contrario se recomienda acudir a alguna de las fuentes bibliográficas básicas empleadas en la redacción de este documento como el excelente libro de David y Alla [4].

En primer lugar se repasarán los conceptos de mayor importancia relacionados con este tipo de redes. A continuación se definirán las ampliaciones que realizaremos al modelo básico para la obtención de un nuevo paradigma para la representación de sistemas híbridos basado en redes de Petri. Por último se estudiarán aplicaciones al control.

2 REDES DE PETRI

2.1 REDES DE PETRI CLÁSICAS

Una Red de Petri es una estructura R= $\langle P, T, Pre, Post, m\rangle$ donde:

P=$\{P_1, P_2, \ldots, P_i\}$ es un conjunto finito y no vacío de lugares.
T=$\{T_1, T_2, \ldots, T_j\}$ es un conjunto finito y no vacío de transiciones.
P$\cap$T=$\emptyset$, es decir, los conjuntos P y T son disjuntos.
Pre: P$\times$T$\rightarrow$R es la aplicación de entrada o, dicho de otra forma, Pre(P_i,T_j) es el "peso" del arco que va desde el lugar P_i a la transción T_j.
Post: P$\times$T$\rightarrow$R es la aplicación de salida. Post(P_i, T_j) es el "peso" del arco que va desde T_j al lugar P_i.
m: P$\rightarrow$ R^i es el marcado inicial.

2.2 REDES DE PETRI HÍBRIDAS

La necesidad de modelar los fenómenos híbridos conduce, como decíamos, a la búsqueda de extensiones a este paradigma de modelado. Las re-

des de Petri híbridas pueden recoger el fenómeno híbrido con ciertas modificaciones al modelo original, aunque a costa de sacrificar la simplicidad y el carácter intuitivo característico de este tipo de redes.

La primera definición de interés de redes de Petri híbridas fue dada por David y Alla en 1987. Estas redes de Petri híbridas se basan en la combinación de redes continuas y discretas, donde sólo los lugares y las transiciones tienen características especiales. Las redes de Petri continuas [4] nos resultan útiles porque ofrecen trabajar en el espacio continuo de estados. El espacio de estado se convierte en infinito de este modo y abre la posibilidad de modelar dinámicas continuas. Lo que estas ampliaciones buscan combinar espacios de estado continuos y discretos para modelar sistemas híbridos.

La simpleza estructural de las marcas en estas redes impone limitaciones que hacen excesivamente difícil el modelado de sistemas complejos. Por esto se definieron las redes de Petri de alto nivel [5] donde las marcas son diferentes entre sí y pueden contener datos estructurados.

2.3 HDN (HYBRID DYNAMICAL NETS) y HON (HYBRID OBJECT NETS)

El modelo de red de Petri híbrida propuesto por David y Alla que no se queda corto dada su imposibilidad de modelar el comportamiento dinámico de sistemas continuos. Dado que las redes de Petri continuas permiten modelar valores reales, sólo pueden ser utilizadas para modelar variables de estado continuas. Por lo tanto, se sugieren las siguientes ampliaciones:

- En las redes dinámicas híbridas (HDN, Hybrid Dinamical Net en adelante), la velocidad de disparo de las transiciones continuas puede darse como una función de cantidades de fichas, lo que permitirá representar dinámicas continuas. Los valores de esta función pueden ser tanto positivos como negativos.

- En HDN la cantidad de fichas puede tomar valores negativos y positivos, lo que permite modelar variables del sistema tanto positivas como negativas. Las HPN sólo permitían valores positivos.

2.4 MODELO DE RED DE PETRI HÍBRIDA PROPUESTO PARA EL SIMULADOR

Ante todo, se ha buscado la simplicidad en el momento de definir los componentes con los que se construirán los modelos. Se ha procurado minimizar el número de comnponentes necesarios hasta reducirlos a un total de 5:

- **Lugar:** se ha definido un único tipo de lugar que tiene asociada una cantidad real que actúa como número de marcas contenidas en el mismo. Cada lugar tiene asociado además un nombre que sirve para referenciar el número de marcas contenidas en su interior. Se representa como un círculo con el número de marcas escrito en su interior.

- **Transición discreta:** este tipo de transición se dispara si se cumple las condiciones impuestas por los arcos de entrada durante un determinado tiempo al que denominaremos *Retraso*. Cada transición discreta tiene dos propiedades asociadas que se pueden editar. La primera es el nombre, que sirve a efectos de identificación. La segunda es la función de retraso, cuyo valor instantáneo proporciona el valor . Se representa con un rectángulo de color negro.

- **Transición continua:** este tipo de transición se encuentra activa siempre que se cumplan las condiciones de *flujo* impuestas por los arcos de entrada. Tiene asociadas dos propiedades que se pueden editar. La primera es el nombre, que sirve a efectos de identificación. La segunda es la función de flujo, cuyo valor instantáneo proporciona el valor de *flujo*. Se representa con un rectángulo de color blanco.

- **Arco:** su función es la de imponer las condiciones de entrada a las transiciones o el resultado de salida de las mismas a través del valor de la propiedad*Peso*. Para ello cuenta con dos propiedades editables. La primera es el nombre, cuya misión es la de identificación, y la segunda es la función peso, cuyo valor instantáneo define la variable . Se representa como una flecha de color negro.

- **Arco inhibidor:** se diferencia del anterior en que establece un tope a través de la propiedad *Peso* que no debe superarse para que se pueda validar la transición hacia la que se dirige. Por tanto, sólo tiene sentido como arco de entrada. Nuevamente, se pueden editar las propiedades Nombre y Función Peso. Se

representa como una linea con un pequeño círculo blanco superpuesto.

Para acceder a la edición de las propiedades de los diversos componentes basta con hacer doble clic sobre ellos. El programa dispone de una opción de guardado y recuperado de archivos xml con estructuras de red de Petri creadas por el programa.

2.4.1 Funciones admisibles

Una de las características especiales del programa que aquí presentamos es la de incluir funciones en los pesos de los arcos, el flujo y el retraso de las transiciones. Las expresiones que las caracterizan pueden emplear las siguientes funciones:

- **SQR(X):** elevar al cuadrado X.
- **SIN(X):** seno de X.
- **COS(X):** coseno de X.
- **ATAN(X):** arcotangente de X.
- **SINH(X):** seno hiperbólico de X.
- **COSH(X:** coseno hiperbólico de X.
- **EXP(X):** exponencial de X.
- **LN(X):** logaritmo neperiano de X.
- **LOG(X):** logaritmo en base 10 de X.
- **LOGN(X,Y):** calcula el logaritmo de Y en base X.
- **SQRT(X):** raíz cuadrada de X.
- **ABS(X):** valor absoluto de X.
- **TRUNC(X):** elimina la parte decimal de X.
- **CEIL(X):** redondea hacia el siguiente entero mayor X.
- **FLOOR(X):** redondea hacia el siguiente entero menor X.
- **RND(X):** genara un número aleatorio entre 0 y X.
- **INTPOW(X,Y):** calcula X^Y, donde Y debe ser un número entero.
- **MAX(X,Y):** devuelve el número mayor de los dos.
- **MIN(X,Y):** devuelve el número menor de los dos.
- **IF(BOOL,X,Y):** si BOOL es diferente de 0, devuelve Y. En caso contrario devuelve X.

En las anteriores expresiones X o Y pueden ser expresiones. El único requisito es que el resultado procedente de evaluarlas sea un número real.

Por si todo este elenco de funciones a nuestra disposición no fuera suficiente, veremos en la próxima sección como usar Matlab para implementar funciones propias.

2.4.2 Comunicación con Matlab

Es evidente que el programa matemático de mayor difusión en el ámbito de la ingeniería es Matlab. El repertorio de herramientas de este programa es tan extenso que resulta muy interesante establecer algún tipo de compatibilidad con el mismo.

Habida cuenta de que el propósito fundamental del programa es el de servir para labores de simulación y control, es requisito que la comunicación pueda establecerse en tiempo real entre ambos programas. Para llevar a cabo esta tarea de comunicación existen dos alternativas básicas: COM (Component Object Model) o DDE (Dynamic Data Exchange). De estas dos, se ha elegido la última opción como medio de comunicación, ya que es la que más facilita la comunicación bidireccional entre MATLAB y el simulador.

Dynamic Data Exchange es una tecnología de comunicación entre varias aplicaciones que puede funcionar en Windows y OS/2. Básicamente, lo que DDE permite es establecer sesiones entre dos aplicaciones y enviar comandos o recibir respuestas en una arquitectura cliente servidor. Se trata de un sistema de comunicación un tanto superado ya por OLE y COM automation. De hecho, Mathworks no realiza nuevos desarrollos para DDE con Matlab desde la versión 5.1. No obstante, es muy sencillo de implementar y la funcionalidad que ofrece es más que suficiente para nuestras necesidades.

En la práctica, la forma en la que ambos programas se integran requiere ejecutarlos en paralelo. Desde un archivo .m de Matlab se pueden ejecutar las siguientes acciones:

ID = ddeinit('Simulador', 'DDETema'): con esta instrucción se establece la comunicación con entre Matlab y el simulador. Es recomendable que éste tenga cargado ya la PN que se simulará.

ddeexec(ID,'ORDEN'): donde ORDEN es una de las siguientes instrucciones:

- SET NOMBREVARIABLE VALOR: establece VALOR como valor de NOMBREVARIABLE.
- GET NOMBREVARIABLE VALOR: lee el

valor de NOMBREVARIABLE.

- START: inicia la simulación del simulador.
- STOP: detiene el simulador.

Con este sencillo juego de instrucciones es suficiente para realizar multitud de aplicaciones de control desde Matlab. La única restricción viene impuesta por el carácter asíncrono del sistema de comunicación empleado, que exige no saturar al servidor DDE con excesivas peticiones. Por tanto, se recomienda esperar un pequeño tiempo entre instrucciones, que irá en función del equipo empleado. A modo indicativo, podríamos exigir 5 centésimas de segundo como periodo mínimo entre instrucciones.

3 EJEMPLO DE APLICACIÓN

Con el fin de demostrar las bondades del simulador realizado, se ha realizado una pequeña demostración. El ejemplo en sí no es excesivamente sofisticado, pero en él se dan cita la mayoría de las características que lo hacen diferente.

El sistema modelado es un cruce regulado por semáforos. Por simplicidad se ha supuesto que los coches circulan en un único sentido en cada uno de las dos vías del cruce.

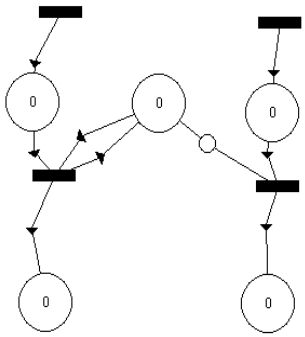

Figura 1: Cruce modelado como red de Petri.

En la figura correspondiente puede observarse que hay dos fuentes que modelan la llegada aleatoria de los vehículos. En concreto se han escogido dos variables aleatorias de tipo rectangular para modelar el número de vehículos que llegan por segundo. Estos vehículos entran en sendos lugares que modelan la cola que se forma ante el semáforo. En el centro de la red hay un lugar que modela el semáforo. Una marca en su interior abre el semáforo para la vía de la izquierda, mientras que la ausencia de ésta lo abre para la de la derecha. Las transiciones responsables del paso por el cruce tienen funciones de retraso asociadas variables en el tiempo, de forma que se tiene en cuenta que la número de vehículos que atraviesa el cruce depende de forma no lineal del tiempo que el semáforo permanece abierto. La dependencia temporal escogida es una exponencial negativa. Finalmente, hay dos lugares que representan la salida de los coches una vez franqueado el cruce.

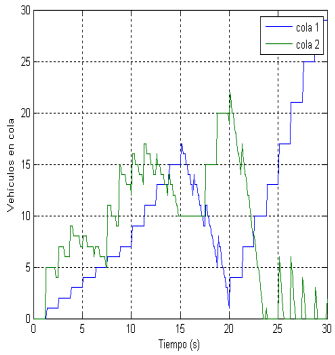

Figura 2: Resultados demo en Matlab.

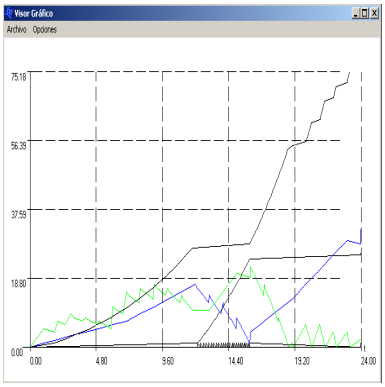

Figura 3: Resultados demo programa.

En la demo se ejecutan dos versiones del sistema presentado, una que se corresponde con el sistema real y otra, con una dinámica 100 veces más rápida, que hace lo propio con el sistema simulado. Decimos versiones porque ambas tienen parámetros de modelado ligeramente diferentes. La idea es emplear los datos de la versión simulada para realizar un control de la real lo más eficiente posible. El control lo realiza el Matlab, que cada 5 segundos estudia la evolución del sistema simulado a partir de los datos proporcionados por el sistema real y calcula de acuerdo con una función de coste la conveniencia o no de tener el semáforo abierto. En cierto modo, estamos aplicando un control predictivo, aunque no tanto por la notación como por el espíritu.

Los resultados de ejecutar la demo se muestran mediante dos figuras, una realizada con Matlab y otra extraída desde el programa realizado, que cuenta con una opción de representación gráfica

de todas las variables que se están manejando. En estas gráficas se representan la cantidad de coches en cola para atravesar el cruce. Matlab decide qué carril permanece abierto a la circulación basándose en las predicciones realizadas en el simulador de redes de Petri. Respecto a la gráfica generada por el propio simulador, cabe destacar que, aunque la subaplicación de representación gráfica no es excesivamente potente, permite al programa proporcionar una solución autocontenida para el trabajo con redes Petri híbridas.

Referencias

[1] Jan Lunze. (2002) What is a Hybrid System?. Modelling, Analysis and Design of Hybrid Systems. Springer Verlag.

[2] John Lygeros.() Lecture Notes on Hybrid Systems.

[3] R Drath.2005 Description of Hybrid Systems by Modified Petri Nets. Modelling, Analysis and Design of Hybrid Systems.

[4] R. David and H. Alla. (2005) Discrete, Continuous and Hybrid Petri Nets. Springer Verlag.

[5] Jensen y Rosenberg. (1991) High Level Petri Nets: Theory and Application. Springer-Verlag, New York.

[6] R. David and H. Alla.(2005) Petri Nets for Modeling of Dynamic Systems- A Survey.Automatica Vol. 30. No 2. pp. 175-202.

[7] Ralf Wieting.(1996) Hybrid High-Level Nets. Proceedings of the 1996 Winter Simulation Conference.

[8] I. Demongodin and N.T. Koussoulas. (1998) Modelling of Hybrid Control Systems using Petri Nets. Proc. 3rd Int. Conf. ADPM'98.